

METHOD AND SYSTEM FOR TRIGGERING A DEBUGGING UNITBACKGROUND OF THE INVENTION

5

Field Of The Invention

The present invention generally relates to triggering a debugging unit, and in particular, a microprocessor configured in accordance with an instruction set architecture for transitioning a debugging unit between a plurality of operating states as directed by trigger instruction signals within an instruction stream.

15 Description Of The Related Art

The arrangement of components on an integrated circuit increases in complexity with each improvement in the manufacturing capability of constructing additional transistors onto smaller chips. Thus, in order to meet market demand, adequate and timely testing and debugging of integrated circuits has become a priority.

Currently, there exist several methods of testing and debugging components of an integrated circuit by controlling and/or monitoring a storage of trace data by a debugging unit. One method includes marking instruction addresses of a sequence of operating instructions that are suspected of generating a problem within the integrated circuit. A storage of trace data commences upon an execution of the suspected operating instructions, and ceases after the execution of the suspected operating instructions. Another method includes marking an operating instruction to commence a storage of trace data upon the execution of the operating instruction, and marking a subsequent operating instruction to cease a storage of trace data upon the execution of the subsequent operating instruction. An additional method includes detecting a particular pattern of trace data being provided via a bus to a trace array. Yet another method includes generating signals internal to a

multi-state logic analyzer for controlling an operation of a trace array in selectively storing trace data.

All of the aforementioned methods of testing and debugging components of an integrated circuit have not always produced
5 consistent and reliable results. The computer industry is therefore continually striving to improve upon the monitoring of trace data by a debugging unit.

SUMMARY OF THE INVENTION

10 The present invention provides a structure and method for placing special triggering instructions only in those selected locations where there is a desire to capture a trace of the failing instruction stream. This is in contrast to the prior art
15 where the marking of general instructions can initiate numerous unintended and undesirable triggers from the processor core to the debugging unit, in that the instructions subject to such marking can occur many times in the instruction stream and not just in the failing case where debugging is desired.

20 One form of the present invention is a method for transitioning a debugging unit between a plurality of operating states. First, operating instructions are defined. The operating instructions are to operate a processing core. Second,
25 a first triggering instruction is defined. The first triggering instruction is to provide a first signal to the debugging unit whereby the debugging unit is operable to transition from a first operating state to a second operating state. Third, the first triggering instruction is embedded within the operating instructions.

30 A second form of the present invention is a microprocessor comprising a debugging unit and a processor core. The debugging unit is operable to transition from a first operating state to a second operating state in response to a first signal. The processor core is operable to fetch an instruction stream
35 including a second signal representative of a first triggering

instruction to transition the debugging unit from the first operating to the second operating state. The processor core is further operable to provide the first signal to the debugging unit in response to the second signal.

5 A third form of the present invention is a computer readable medium comprising a first computer readable code and a second computer readable code embedded within the first computer readable code. The first computer readable code is to operate a processor core. The second computer readable code is to
10 transition a debugging unit from a first operating state to a second operating state.

A fourth form of the present invention is a system for transitioning a debugging unit between a plurality of operating states. The system comprises a computer readable medium and a processor core. The computer readable medium is operable to provide a first signal representative of a first computer readable code to transition a debugging unit from a first operating state to a second operating state. The processor core is operable to provide a second signal to the debugging unit in
15 response to the first signal whereby the debugging unit is operable to transition from the first operating state to the second operating state.

The foregoing and other features and advantages of the invention will become further apparent from the following
20 detailed description of the presently preferred embodiments, read in conjunction with the accompanying drawings. The detailed description and drawings are merely illustrative of the invention rather than limiting, the scope of the invention being defined by the appended claims and equivalents thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

30 **FIG. 1** is a block diagram of a first embodiment of a microprocessor and a computer readable medium in accordance with the present invention;

FIG. 2A is a block diagram of a first embodiment of testcase in accordance with the present invention;

FIG. 2B is a block diagram of a second embodiment of a microprocessor in accordance with the present invention;

5 **FIG. 3A** is a block diagram of a second embodiment of a testcase in accordance with the present invention;

FIG. 3B is a block diagram of a third embodiment of a microprocessor in accordance with the present invention;

10 **FIG. 4A** is a block diagram of third embodiment of a testcase in accordance with the present invention; and

FIG. 4B is a block diagram of a fourth embodiment of a microprocessor in accordance with the present invention.

DETAILED DESCRIPTION

15 Referring to **FIG. 1**, a microprocessor **10** in accordance with the present invention is shown. Microprocessor **10** includes a processor core **20**, and a debugging unit **30**. Processor core **20** is a compilation of circuitry for fetching, decoding, and executing an instruction stream **IS** of operating signals from a main memory **41** and/or a cache **42** of computer readable medium **40**. Processor core **20** provides trace data **TRD** to debugging unit **30** as the operating signals of instruction stream **IS** are being processed by processor core **20**. Debugging unit **30** is a state machine for
20 selectively storing trace data **TRD** within an internal memory component. The present invention configures processor core **20** and computer readable medium **40** in accordance with an instruction set architecture of the present invention that enables processor core **20** to provide a trigger event signal **TE_{s1}**, a trigger event signal **TE_{s2}**, and/or a trigger event signal **TE_{s3}** to debugging unit
25 **30** in response to trigger instruction signals within instruction stream **IS**. For purposes of the present invention, a triggering instruction signal is defined as a non-operative signal, i.e. the architecture state of processor core **20** does not change in
30 response to the triggering instruction signal. This is to be distinguished from an operating instruction that changes the
35

architecture state of processor core 20 as processor core 20 executes the operating instruction. Debugging unit 30 transitions to a base operating state in response to trigger event signal TE_{s1} , i.e. a reset signal. Debugging unit 30 transitions to an operating state for dynamically storing trace data **TRD** within its internal memory component in response to trigger event signal TE_{s2} , i.e. a start signal to write trace data **TRD** into the internal memory. Debugging unit 30 transitions to an operating state for statically storing trace data **TRD** within its internal memory component in response to trigger event signal TE_{s3} , i.e. a stop signal to hold trace data **TRD** previously written into the internal memory.

In other embodiments of the present invention, debugging unit 30 can be omitted from microprocessor 10, and an electrical communication can be established between microprocessor 10 and an external logic analyzer as would occur to one skilled in the art. In yet other embodiments of the present invention, a central processing unit having processor core 20 or portions thereof, and/or debugging unit 30 or portions thereof formed by multiple integrated circuits can be substituted for microprocessor 10.

Referring to **FIGS. 2A and 2B**, a testcase 150 and a microprocessor 110 in accordance with an instruction set architecture of the present invention is shown. Testcase 150 includes operating instructions 160, a triggering instruction 170, a triggering instruction 171, and a triggering instruction 172. Operating instructions 160 is for operating a processor core 120 of microprocessor 110. Triggering instruction 170 is for transitioning a debugging unit 130 of microprocessor 110 to a base operating state. Triggering instruction 171 is for transitioning debugging unit 130 to an operating state whereby trace array 131 dynamically stores trace data **TRD** from a processor core 120 of microprocessor 110 (hereinafter "the dynamic storing operating state"). Triggering instruction 172 is for transitioning debugging unit 130 to an operating state whereby trace array 131 statically stores trace data **TRD**

(hereinafter "the static storage operating state"). Triggering instruction 170, triggering instruction 171, and triggering instruction 172 are strategically embedded within operating instructions 160 transition debugging unit 130 between the base operating state, the dynamic storage operating state, and the static storage operating state.

Testcase 150 is coded within main memory 41 (FIG. 1) or cache 42 (FIG. 1). Processor core 120 fetches an instruction stream IS_1 including operating signals (not shown) that are representative of operating instructions 160, a trigger instruction signal TI_{s1} that is representative of triggering instruction 170, a trigger instruction signal TI_{s2} that is representative of triggering instruction 171, and a trigger instruction signal TI_{s3} that is representative of triggering instruction 172.

Processor core 120 includes a register 122, a register 123, and a register 124. Register 122, register 123, and register 124 are shown as being separate from processor core 120 to simplify the description of processor core 120.

Processor core 120 provides a register address signal RA_{s1} to register 122 in response to trigger instruction signal TI_{s1} . Register 122 provides trigger event signal TE_{s1} (FIG. 1) to debugging unit 130 in response to register address signal RA_{s1} . A logic analyzer 132 of debugging unit 130 transitions debugging unit 130 to the base operating state in response to trigger event signal TE_{s1} .

Processor core 120 provides a register address signal RA_{s2} to register 123 in response to trigger instruction signal TI_{s2} . Register 123 provides trigger event signal TE_{s2} (FIG. 2) to debugging unit 130 in response to register address signal RA_{s2} . Logic analyzer 132 transitions debugging unit 130 to the dynamic storage operating state in response trigger event signal TE_{s2} . Specifically, logic analyzer 132 provides a write enable signal WE_s to trace array 131 in response to trigger event signal TE_{s2} .

Trace array 131 dynamically store trace data TRD in response to write enable signal WE_s.

Processor core 120 provides a register address signal RA_{s3} to register 124 in response to trigger instruction signal TI_{s3}.

5 Register 124 provides trigger event signal TE_{s3} (FIG. 1) to debugging unit 130 in response to register address signal RA_{s3}. Logic analyzer 132 transitions debugging unit 130 to the static storage operating state in response trigger event signal TE_{s2}. Specifically, logic analyzer 132 ceases any provision of write
10 enable signal WE_s to trace array 131 in response to trigger event signal TE_{s3}. Trace array 131 statically stores any trace data TRD written into trace array 131 during the dynamic storage operating state.

It is to be appreciated that the processing of trigger
15 instruction signal TI_{s1}, trigger instruction signal TI_{s2}, and trigger instruction signal TI_{s3} by processor core 120 transitions debugging unit 130 between the base operating state, the dynamic storage operating state, and the static storage operating state. Consequently, upon the completion of processing instruction
20 stream IS₁ by processor core 120, the trace data TRD stored within trace array 131 is representative of the results of processing portions of testcase 150 by processor core 120.

Referring to FIGS. 3A and 3B, a testcase 151 and a
25 microprocessor 111 in accordance with an instruction set architecture of the present invention is shown. Testcase 151 includes operating instructions 160 (FIG. 2A), triggering instruction 170 (FIG. 2A), a set of operating instructions 173, triggering instruction 171 (FIG. 2A), and triggering instruction 172 (FIG. 2A). Operating instructions 173 are for generating
30 trigger data or non-event data. Triggering instruction 170 and triggering instruction 172 are strategically embedded within operating instructions 160 to transition debugging unit 130 to the base operating state and the static storage operating state, respectively. Operating instructions 173 and triggering
35 instruction 172 are sequentially and strategically embedded

within operating instructions **160** to optionally transition debugging unit **130** to the dynamic storage operating state.

Testcase **151** is coded within main memory **41** or cache **42** (**FIG. 1**). Processor core **120** fetches an instruction stream **IS₁** including operating signals (not shown) that are representative of operating instructions **160**, trigger instruction signal **TI_{s1}** (**FIG. 2B**), trigger instruction signal **TI_{s2}** (**FIG. 2B**), trigger instruction signal **TI_{s3}** (**FIG. 2B**), and a data instruction signals **DI_{s1}** that is representative of operating instructions **173**.

Processor core **120** includes register **122** (**FIG. 2B**), register **124** (**FIG. 2B**), and a register **125**. Register **122**, register **124**, and register **125** are shown as being separate from processor core **120** to simplify the description of processor core **120**.

Processor core **120** provides register address signal **RA_{s1}** to register **122** in response to trigger instruction signal **TI_{s1}**. Register **122** provides trigger event signal **TE_{s1}** (**FIG. 1**) to debugging unit **130** in response to register address signal **RA_{s1}**. Logic analyzer **132** transitions debugging unit **130** to the base operating state in response to trigger event signal **TE_{s1}**.

In response to data instruction signals **DI_{s1}**, processor core **120** provides either a trigger data signal **TD_{s1}** to register **25** when processor core **120** generates trigger data, or provides a non-event data signal **ND_{s1}** to register **25** when processor core **120** generates the non-event data. For example, processor core **120** can perform a XOR operation of two general purpose registers (not shown) in response to data instruction signals **DI_{s1}**. The contents of one registers can be a pre-defined constant. The contents of the other register can be a testcase number for test case **151** that matches the pre-defined constant, or any other number. Trigger data can be defined as the result of a match of the pre-defined constant and the testcase number for testcase **151**, i.e. the XOR operation yielding all zeros. Non-event data can be defined as the results of a mismatch of the pre-defined constant and any other number, i.e. the XOR operation yielding some ones.

Subsequent to a provision of either trigger data signal TD_{s1} or non-event data signal ND_{s1} by processor core 120, processor core 120 provides register address signal RA_{s2} to register 125 in response to trigger instruction signal TI_{s2} . Register 125

5 provides trigger event signal TE_{s2} (FIG. 1) to debugging unit 130 in response to register address signal RA_{s2} and trigger data signal TD_{s1} . Logic analyzer 132 transitions debugging unit 130 to the dynamic storage operating state in response to trigger event signal TE_{s2} .

10 Register 125 does not provide trigger event signal TE_{s2} (FIG. 1) to debugging unit 130 in response to register address signal RA_{s2} and no-event data signal ND_{s1} .

Sub Processor core 120 provides register address signal RA_{s3} to register 124 in response to trigger instruction signal TI_{s3} .

Fig Register 124 provides trigger event signal TE_{s3} (FIG. 1) to debugging unit 130 in response to register address signal RA_{s3} . Logic analyzer 132 transitions debugging unit 130 to the static storage operating state in response trigger event signal TE_{s2} .

Sub It is to be appreciated that the processing of trigger instruction signal TI_{s1} , data instruction signals DI_{s1} , trigger instruction signal TI_{s2} , and trigger instruction signal TI_{s3} by processor core 120 transitions debugging unit 130 to the base operating state and the static storage operating state, and selectively transition debugging unit 130 to the dynamic storage operating state. Consequently, upon the completion of processing instruction stream IS_1 by processor core 120, any trace data TRD stored within trace array 131 is representative of the results of processing portions of testcase 151 by processor core 120.

Sub Referring to FIGS. 4A and 4B, a testcase 152 and a microprocessor 112 in accordance with an instruction set architecture of the present invention is shown. Testcase 152 includes operating instructions 160 (FIG. 2A), triggering instruction 170 (FIG. 2A), a set of operating instructions 174, triggering instruction 171 (FIG. 2A), and triggering instruction 172 (FIG. 2A). Operating instructions 174 are to generate a

first trigger data or a second trigger data. Triggering instruction 170 and triggering instruction 172 are strategically embedded within operating instructions 160 to transition debugging unit 130 to the base operating state and the static storage operating state, respectively. Operating instructions 174 and triggering instruction 172 are sequentially and strategically embedded within operating instructions 160 to selectively transition debugging unit 130 to the dynamic storage operating state or the base operating state.

Testcase 152 is coded within main memory 41 or cache 42 (FIG. 1). Processor core 120 fetches an instruction stream IS_3 including operating signals (not shown) that are representative of operating instructions 160, trigger instruction signal TI_{S1} (FIG. 2B), a data instruction signal DI_{S2} , trigger instruction signal TI_{S2} (FIG. 2B), and a trigger instruction signal TI_{S3} (FIG. 2B).

Processor core 120 includes register 122 (FIG. 2B), register 124 (FIG. 2B), and a register 126. Register 122, register 124, and register 126 are shown as being separate from processor core 120 to simplify the description of processor core 120.

Processor core 120 provides register address signal RA_{S1} to register 122 in response to trigger instruction signal TI_{S1} . Register 122 provides trigger event signal TE_{S1} (FIG. 1) to debugging unit 130 in response to register address signal RA_{S1} . Logic analyzer 132 transitions debugging unit 130 to the base operating state in response to trigger event signal TE_{S1} .

In response to data instruction signals DI_{S2} , processor core 120 provides either a trigger data signal TD_{S2} to register 126 when processor core 120 generates the first trigger data, or provides a trigger data signal TD_{S2} to register 126 when processor core 120 generates the second trigger data. Subsequent to a provision of either trigger data signal TD_{S2} or trigger data signal TD_{S3} by processor core 120, processor core 120 provides register address signal RA_{S2} to register 126 in response to trigger instruction signal TI_{S2} . Register 126 provides trigger

event signal **TE_{s2}** (**FIG. 1**) to debugging unit **130** in response to register address signal **RA_{s2}** and trigger data signal **TD_{s2}**. Logic analyzer **132** transitions debugging unit **130** to the dynamic storage operating state in response to trigger event signal **TE_{s2}**. Register **126** provides trigger event signal **TE_{s3}** (**FIG. 1**) to debugging unit **130** in response to register address signal **RA_{s2}** and trigger data signal **TD_{s3}**. Logic analyzer **132** transitions debugging unit **130** to the static storage operating state in response to trigger event signal **TE_{s3}**.

Processor core **120** provides register address signal **RA_{s3}** to register **124** in response to trigger instruction signal **TI_{s3}**. Register **124** provides trigger event signal **TE_{s3}** (**FIG. 1**) to debugging unit **130** in response to register address signal **RA_{s3}**. Logic analyzer **132** transitions debugging unit **130** to the static storage operating state in response trigger event signal **TE_{s2}**.

It is to be appreciated that the processing of trigger instruction signal **TI_{s1}**, trigger instruction signal **TI_{s3}**, trigger instruction signal **TI_{s6}**, and trigger instruction signal **TI_{s7}** by processor core **120** transitions debugging unit **130** to the base operating state and the static storage operating state, and selectively transition debugging unit **130** to either the dynamic storage operating state or the base operating state. Consequently, upon the completion of processing instruction stream **IS₃** by processor core **120**, any trace data **TRD** stored within trace array **131** is representative of the results of processing portions of testcase **152** by processor core **120**.

From the previous descriptions of the present invention in connection with **FIGS. 2A-4B**, one skilled in the art will know how to make and use other embodiments of test cases and microprocessors in accordance with the present invention. For example, one skilled in the art will know how to make and use a test case including one or more triggering instructions **170** (**FIG. 2A**); one or more triggering instructions **171** (**FIG. 2A**); one or more triggering instructions **172** (**FIG. 2A**); one or more sets of operating instructions **173** (**FIG. 3A**); and/or one or more sets of

operating instructions 174 (FIG. 4A). Also by example, one skilled in the art will know how to make and use a microprocessor including one or more registers 122 (FIG. 2B); one or more registers 123 (FIG. 2B); one or more registers 124 (FIG. 2B); one or more registers 125 (FIG. 3B); and/or one or more registers 126 (FIG. 3B).

Thus, the present invention provides a structure and method for placing special triggering instructions only in those selected locations where there is a desire to capture a trace of the failing instruction stream. This is in contrast to the prior art where the marking of general instructions can initiate numerous unintended and undesirable triggers from the processor core to the debugging unit, in that the instructions subject to such marking can occur many times in the instruction stream and not just in the failing case where debugging is desired.

Though the invention has been described in the context of a uniprocessor core, the underlying concepts as claimed herein are equal applicable and beneficial in a multiprocessor system with multiple individual processor cores.

While the embodiments of the present invention disclosed herein are presently considered to be preferred, various changes and modifications can be made without departing from the spirit and scope of the invention. The scope of the invention is indicated in the appended claims, and all changes that come within the meaning and range of equivalents are intended to be embraced therein.